# Java

1. Java - Обектно-ориентиран език за програмиране на високо ниво. Няма директен достъп до данни, а се достъпват през предвиден за целта код.
    a. Създаден от Sun Microsystems и James Gosling през 1995 г.
    b. Повлиян от C++, Smalltalk и Objective-C
    c. Java 17 има 4859 класа и интерфейси
    d. Програми
        - 2 вида - приложения и аплети
        - Могат да съдържат повече от един клас
        - При приложенията точно един клас трябва да съдържа дефиниция на метод с име main (който се активира при изпълнението на програмата)
2. Основни конструкции
    a. if-else
    b. switch
    c. while
    d. do-while
    e. for

3. Обектно-ориентирано програмиране - вид парадигма в компютърното програмиране.
    a. Програмната система се моделира като набор от обекти, които взаимодействат помежду си, за разлика от традиционното виждане, в което една програма е списък от инструкции, които компютърът изпълнява.
    b. Всеки обект е способен да получава съобщения, обработва данни и праща съобщения на други обекти.
    c. Фундаментални ООП принципи
        - Енкапсулация - скриване на данните на обекта.
        - Наследяване - техника или процес, при който един обект от клас придобива поведението и свойствата на друг обект. Това се прави чрез наследяване на класа или установяване на връзка между два класа.
        - Полиморфизъм - способността на даден обект да се държи като инстанция на друг клас или като имплементация на друг интерфейс. Наследниците на даден клас споделят поведение от родителския клас, но могат да дефинират и собствено поведение.
            - Всички Java обекти са полиморфични, защото всеки обект наследява java.lang.Object класа

- Абстракция - когато се моделира обект, да се ограничава само до съществените характеристики и да се абстрахира (пропуснат) несъществените. Абстракция също значи работа с нещо, което знаем как да използваме, без да знаем как работи.

d. Ограничение на достъп
   - Капсулиране включва скриване на подробностите за това как работи даден софтуер.
   - public - може да бъде извикан извън класа.
   - private (частен) - не може да бъде извикан извън дефиницията на класа.
     - Всички променливи на обект от клас се правят частни, за да се ограничи използването им само чрез методите.
     - Оперира като стена между създателя и клиента.
   - protected - може да бъде извикан само от класове, които наследяват класа в който е дефиниран.

e. Обекти - множество данни, които могат да се предават и достъпват само чрез методи.
   - Данните съставляват състоянието на обекта.
   - Методите съставляват поведението му.
   - Състоянието на обекта е скрито (енкапсулирано) от директен достъп.
   - Обектът е инстанция (представител на даден клас. Обекти се създават явно (с оп. new) или неявно и се съдържат в heap паметта.
   - Обект може да се създаде по два начина - по имплицитен и експлицитен. Имплицитния оставя виртуалната машина да реши типа, а при експлицитния се определя от програмиста.
   - Ключова дума this - референция към конкретния обект.
     - Употребява се за достъпване на променливи

f. Конструктор - специален метод за създаване на нов обект (изпълнение на инициализации).
   - Конструктор по подразбиране - Конструктор без параметри

g. Класове - дефиниция на обекти (или клас от такъв)
   - Описва състояние чрез данни
   - Описва поведение чрез методи
   - Възможно е да няма състояние/поведение
   - Всеки клас има определен интерфейс

h. Метод - функция за манипулиране на състоянието на клас
   - Сигнатура - име и списък от параметри
   - Арност - брой на параметри
   - Два метода имат еднаква сигнатура, ако имат еднакви имена, арност и последователност от типове в списъка от параметри

- Method overriding - класът - наследник предефинира поведението на класа-родител.
- Method overloading - класът декларира методи с едно и също име и различен брой параметри.
- Overloading има по-добър runtime performance от overriding.

| | Overloading | Overriding |
|---|---|---|
| Кога | Compile-time | Runtime |
| Къде | В същия клас | В наследниците |
| Списък с аргументи | Различен | Идентичен |
| Return type | Може да е различен | Съвместим |
| static, private и final | Да | Не |

- Статични методи имат достъп само до статичните променливи и други статични методи на класа
    - Нестатичните методи имат достъп като до статичните, така и до нестатичните членове на класа.
    - Те са част от класа, а не от негов обект.
    - Могат да се достъпват и без да е създаден обект.

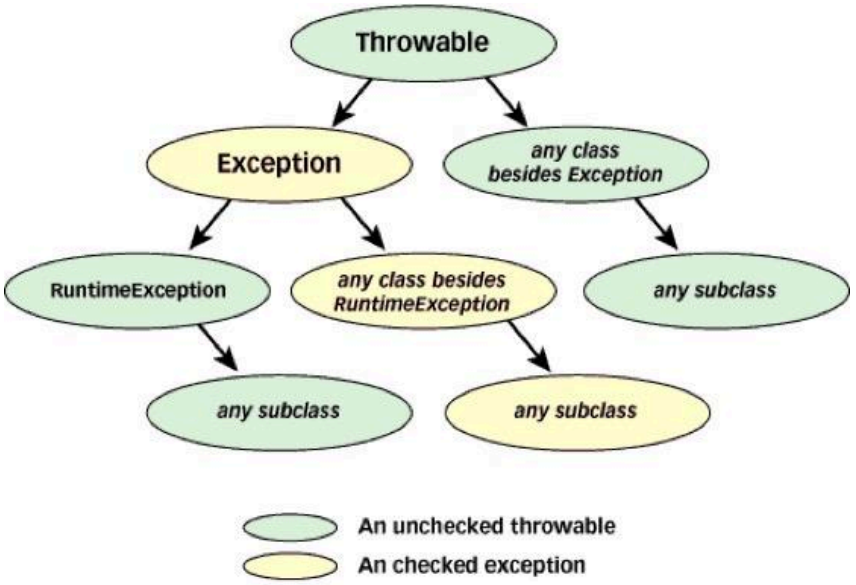
i. Абстрактни класове - дефинират се с модификатора abstract.
- Могат да  имат методи без имплементация, които се дефинират с метода abstract.
- Не са напълно дефинирани.
- Абстрактните класове не могат да имат final.

j. Интерфейс - съвкупност от декларации на методи без имплементация. Описват формално поведение без да го имплементират.
- Може да наследява множество интерфейси.
- Модификатори - public и abstract по подразбиране.
- Default методи - метод, който има имплементация и модификатор default в декларацията си.
- Имплементиращите класове имплицитно ползват default-ната имплементация на методите, но могат и да я предефинират.
- Ако два или повече от тях съдържат default метод с еднаква сигнатура, класът трябва задължително да предефинира този метод
- В предефинирания метод може експлицитно да се укаже, default-ната имплементация от кой родителски интерфейс да се ползва. В този случай, синтаксисът е, <имеНаИнтерфейс>.super.<имеНаDefaultМетод>()

- Интерфейс, който не съдържа нито един метод, се нарича маркерен.
- Интерфейс, който има точно един публичен абстрактен метод, се нарича функционален.

4. Императивно програмиране
   a. Алгоритъм - Подход за изчисляване на търсени стойности от дадени такива. Използва постъпково изпълнение на елементарни обработващи операции.
   b. Базови концепции
      - Променливи: притежават стойности, които се променят посредством прилагане на оператори
      - Оператори: Служат за достъп до стойностите на променливите (четене, промяна, извеждане на стойностите)
   c. Базов метод за структуриране на императивното програмиране
      - Процедури (функции, методи): Частични алгоритми, представени като оператори на езика
      -
5. Пакети - колекция от класове, които са групирани в папка.
   a. Импорт (import) - предоставя механизъм за управление на “пространства от имена”
   b. Йерархия - package alpha.beta.gamma => “.../alpha/beta/gamma”

6. Изключения - събитие, което се случва по време на изпълнение на дадена програма и нарушава нормалната последователност на изпълнение на инструкциите
   a. Друг начин за комуникация на метод с извикващите го: връщана стойност при нормално изпълнение и изключение при проблем.
   b. Видове изключения -
      - Checked (Compile-time) exceptions - наричат се compile-time exceptions, защото компилаторът задължава да се обработят.
        - Примери: FileNotFoundException, IOException
      - Unchecked (Runtime) exceptions - възникват по време на изпълнение на приложението.
        - Често резултат от семантични/логически грешки в кода.
        - Примери: ArrayIndexOutOfBoundsException, NullPointerException, ClassCastException
      - Errors - проблеми, възникващи извън приложението/не могат да бъдат очаквани.
        - Обикновено се генерират от самата виртуална машина.
        - Примери: OutOfMemoryError, StackOverflowError




7. Многонишково програмиране (Multithreading)
    a. Concurrency - способността да се прави повече от едно нещо в един и същи момент.
    b. Units of Concurrency
        - Multiprocessing - много процесори, изпълняващи инструкции едновременно. Единицата е процесор.
        - Multitasking (Многозадачност) - Много задачи, изпълняващи се едновременно в един процесор. Единицата е процес.
        - Multithreading (Многонишковост) - множество части на една и съща програма се изпълняват едновременно. Единицата е thread (нишка).
    c. Процеси и нишки
        - Процес - програма по време на изпълнение
            - Има си собствено адресно пространство, call stack и handles към ресурси.
        - Нишка - път на изпълнение в процес
            - Всеки процес има поне една нишка, наричана main
            - Всяка нишка може да създава допълнителни нишки
            - Нишките в един процес споделят ресурсите му. Всяка нишка си има собствен call stack.

| | Процеси | Нишки |
|---|---|---|
| Стартиране | Бавно | Относително бързо |
| Изолация | Да | Не |
| Комуникация | Бавна | Бърза |

8. Компилация

a. Байткод - силно оптимизиран набор от инструкции, проектирани да бъдат изпълнявани от системата Java Runtime, която се нарича Java Virtual Machine.
    - Ключът, който позволява на Java да адресира както проблемите със сигурността, така и преносимостта е, че изходът на компилатор на Java не е изпълним код.
    - Write Once Run Everywhere (WORE)
b. Java Virtual Machine (JVM) - абстрактна изчислителна машина, позволяваща да се изпълняват програми написани на определени езици. Осигурява независимостта от хардуер и операционна система.

9. Софтуер
    a. Свойства на софтуера
        - Не се изхабява
        - Лесен за копиране
        - Комплексен
        - Остарява
        - Трудно измерим (качество/количество)
        - Дълго време в употреба
    b. Модели
        - Водопаден
        - Жизнен цикъл
        - Спирален
    c. Критерии за софтуерни продукти
        - Коректност
        - Стабилност (обработка на грешки)
        - Ефективност
        - Удобство за потребителя
        - Документираност
        - Модифицируемост
        - Читаемост (разбираемост)
        - Многократна използваемост
        - Модулност
        - Преносимост
        - Интегрируемост

Композиция:

- основен подход, многократно използване на код, в нов клас се поставят **обекти от вече съществуващи класове** като негови членове